Predicting Neuron Types in the *Drosophila* Brain Wiring Diagram

EE-452 Network Machine Learning — Final Project

Ernesto Bocini (359541), Sibo Wang (320354)

***Abstract*—A complete wiring diagram of neurons, known as a *connectome*, has recently been constructed for the brain of *Drosophila melanogaster*, the common fruit fly [1]. This dataset allows us to gain insights into the interactivity among neurons in the brain, a central question in neuroscience. In the exploration part of this project, we show that the brain connectivity is very sparse and resembles a scale-free graph. In the exploitation stage, we aim to investigate to what extent information about a neuron can be predicted purely from its interconnectivity with other neurons. We demonstrate that, using Graph Neural Networks (GNNs), the neuron type and the hemilineage can be reliably predicted from the connectivity of the graph.**

I. Introduction

A. Background

Connectomics is a discipline within neuroscience that is focused on mapping out the connections of neurons. Recently, a connectome has been reconstructed for the brain of the fruit fly *Drosophila melanogaster* [1], a common model organism in neuroscience. The study of *Drosophila* neuronal circuits is known for being highly tractable, with only about 10^5 neurons in the fly brain (compared to about 10^7 in the mouse brain and about 10^{11} in the human brain [2]). This makes the study of the *Drosophila* brain wiring particularly valuable for the mechanistic understanding of neuronal circuitry.

To reconstruct the *Drosophila* brain connectome, a nanometer-scale 3D image dataset of the brain was first acquired in [3]. In this process, a brain tissue is dissected, stained, and sliced into numerous 40nm-thin sections. Each section is then recorded using a Transmission Electron Microscopy (TEM) at 4nm-by-4nm resolution. The images are stitched together and aligned to create a 3D image dataset. The actual wiring diagram is then derived from this image dataset in [1]: first, the volumetric meshes of the neurons are constructed using computer-vision-based segmentation models. These computer-generated meshes are then exhaustively proofread and edited by human contributors. Synapses (junctions between two neurons where signaling takes place through the transmission of neurotransmitters) and the type of the neurotransmitter are also identified. This allows for the reconstruction of edges in the wiring diagram (*ie.* the interaction between neurons). This process is schematically shown in Fig. 1.

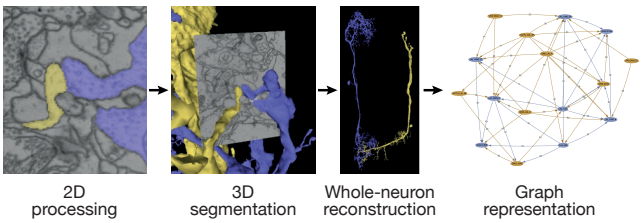

Figure 1. Overview of the processing pipeline for the connectome dataset (panels adapted from https://flywire.ai/).

B. Graph Formulation and Node Embedding

In this section, we explain how we constructed a graph from the raw data from [1]. We frame the *Drosophila* connectome as a multi-layer weighted directed graph $G(V, E, L)$ where V is the set of nodes (each being a neuron in the brain), E is the set of edges (each being a pair of connected neurons), and L is the set of layers (each being a neurotransmitter, *ie.* a distinct chemical that transmits from one neuron to another). Each edge $e \in E$ is a quadraple (u, v, c, w) where $u, v \in V, c \in L$, and the edge weight $w \in \mathbb{N}$ is the number of synaptic sites where neuron u synapses onto neuron v via neurotransmitter c. It is believed that each neuron typically releases a single neurotransmitter [4]; therefore the source nodes in each layer are generally different. Six neurotransmitters are present in our dataset, including three main neurotransmitters: acetylcholine, GABA, glutamate, and three neuromodulators: serotonin, dopamine, octopamine.

Next, we build an low-dimensional embedding for each node summarizing its neighborhood information. We do this independently for each layer to derive a d-dimensional feature vector per layer, and then concatenate them to obtain a $(d \times |L|)$-dimensional embedding for the whole graph. This process is schematically shown in Fig. 2. We built the embedding for each layer using two methods:

- *Spectral graph analysis*: we take the eigenvectors corresponding to the $d = 8$ smallest eigenvalues of the directed Laplacian matrix [5].
- *Node2Vec*: we generate a $d = 8$-dimensional Node2Vec [6] feature using 50 random walks of length 30.

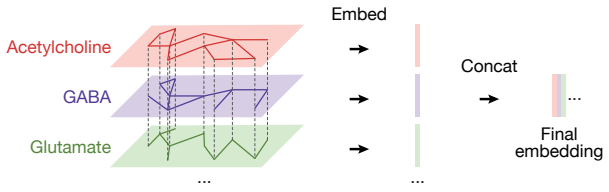

Figure 2. Overview of the node embedding process for the multi-layer graph (only three of six layers shown).

Each neuron also carries a number of node attributes, in particular:

- *Morphological attributes*: skeleton length, surface area, volume.
- *Neuron classifications*: superclass, class, subclass (defined at different hierarchical levels). This label is incomplete in our dataset.
- *Hemilineage*: the lineage of neurons wherein all neurons originate from the same stem cell during development. This label is incomplete in our dataset.

In this project, the morphological attributes are treated as additional node features (the total number of node features is therefore $d \times |L| + 3$). The neuron classifications and the hemilineage are treated as target variables predicted by the Graph Neural Network (GNN) model.

Finally, we exclude all neurons in the optical lobes where images received by the eyes are processed. These neurons are highly specialized sensory neurons that follow a repeated pattern. Critically, these neurons have not been fully proofread, meaning that they are not as accurately reconstructed as neurons in other brain regions.

II. Exploration

A. Node and Edge Statistics

There are 51,405 nodes and 1,301,936 edges in our graph.[1] All neurons carry a valid superclass label; 41.29% of the neurons carry a valid class label; 59.24% of the neurons carry a valid hemilineage label. There are 8 distinct superclasses, 28 distinct classes, and 188 distinct hemilineages. The distributions of the neuron superclasses, classes, and hemilineages are shown in Fig. 3(a, b, c). Most edges (97.11%) in our graph correspond to one of the main neurotransmitters (acetylcholine, GABA, glutamate), with the rest corresponding to the neuromodulators (serotonin, dopamine, octopamin). The distribution of the edge types is shown in Fig. 3(d).

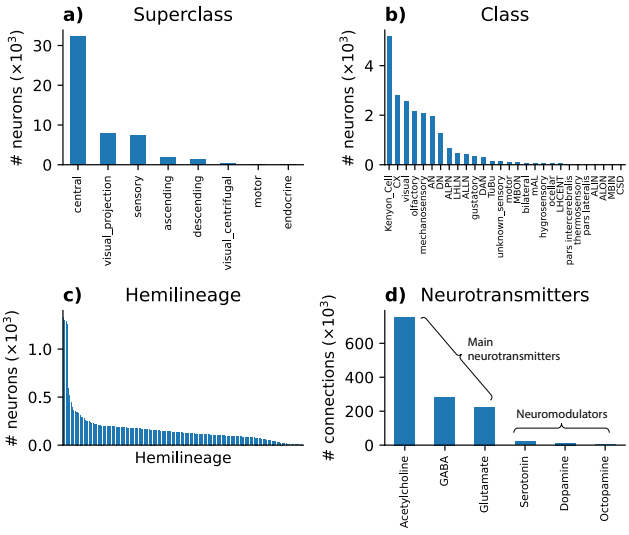

Figure 3. **(a)** The distribution of neuron superclasses. **(b)** The distribution of neuron classes. **(c)** The distribution of neuron hemilineages. **(d)** The distribution of neurotransmitters in the edges.

[1]We use the April 2023 version (snapshot 630) of the FlyWire dataset with all neurons with the superclass "optic" removed, as discussed earlier.

B. Degree Distribution

As observed in the node and edge statistics, the network is very sparse. The density of this graph is 4.5×10^{-4}, defined as

$$\text{Density} = \frac{M}{N(N-1)} \tag{1}$$

where M is the number of (directed) edges and N is the number of nodes. The low density of the graph resembles many real networks with the number of edges scaling with $\mathcal{O}(N)$.

The in- and out-degree distribution, visualized in Fig. 4, shows a power-law scaling. By looking at the green and red dotted curves, our distributions seem to be at the boundary between anomalous ($\alpha \leq 2$) and scale-free regime ($2 < \alpha \leq 3$). However, since the first moment of the in-degree and out-degree distributions is relatively small (respectively 22.11 and 22.10) while the second moment is extremely large (2,636.91 and 2,082.03), the distribution is more likely to be scale-free, for which $\langle k \rangle$ is finite and $\langle k^2 \rangle$ diverges as the number of nodes grows to infinity. Confirming this, we observe many small degree nodes, not so many around the average degree, and a few large degree nodes. In other words, the power law implies that the vast majority of nodes have very few connections, while a few important nodes (we call them Hubs) have a huge number of connections.

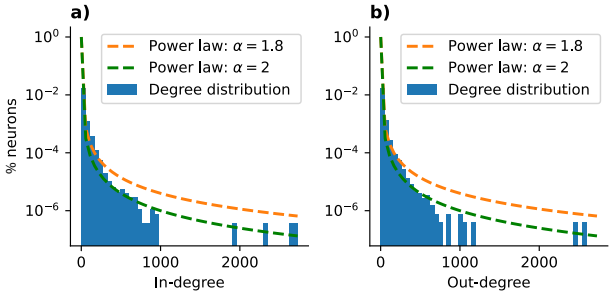

Figure 4. The distributions of **(a)** input and **(b)** output degrees and the power-law fittings.

Understanding the scale-free nature of the graph is important for graph machine-learning tasks. It suggests that some nodes may have a disproportionately higher influence or importance in the overall network structure. This knowledge can be leveraged to design efficient algorithms for tasks such as node classification, link prediction, or community detection.

C. Other Properties of the Graph

Besides the degree distribution, when exploring a graph for machine learning purposes, there are several other properties to check to gain a better understanding of its structure and characteristics. Among them, we highlight the following:

- Global Clustering Coefficient = 0.055
- Average Clustering Coefficient = 0.144
- Average distance $\simeq$ 4.21

The global clustering coefficient, also known as the transitivity, measures the prevalence of triangles in the entire graph. The average clustering coefficient, on the other hand, calculates the local clustering coefficient for each individual node in the graph and then computes the average across all nodes. While the global clustering coefficient provides a measure of clustering in the entire graph, the average clustering coefficient gives insights into the local connectivity of individual nodes. The combination of a low density, low global clustering coefficient, and a higher average clustering coefficient indicates a graph with sparse connections overall, but with localized clusters or neighborhoods exhibiting higher levels of clustering.
As for the average shortest path (i.e., distance), since the graph is disconnected, its true value would be infinity, which however doesn't provide any further insight besides telling that the network is in fact disconnected. Therefore, we decided to compute a proxy for it, by taking the average distance only over the largest connected component inside the graph. Comparing its value, 4.21, to the expected average distance under the assumption of small-world property: $\log(N)/\log\langle k\rangle = 3.50$, we see that they are very close, hinting that the graph may also present small-world properties.

D. Node Clustering

We used t-SNE [7] to visualize the clustering of the neurons using both spectral and Node2Vec features (Fig. 5). We only used the node features on the main neurotransmitter layers (acetylcholine, GABA, glutamate). We observe that distinct clusters of various sizes emerge from the node embedding. Nonetheless, naive unsupervised clustering fails to separate different neurons with different attributes (*ie.* superclasses, classes, hemilineages). We will try to predict the neuron attributes using GNN models in the exploration stage of the project.

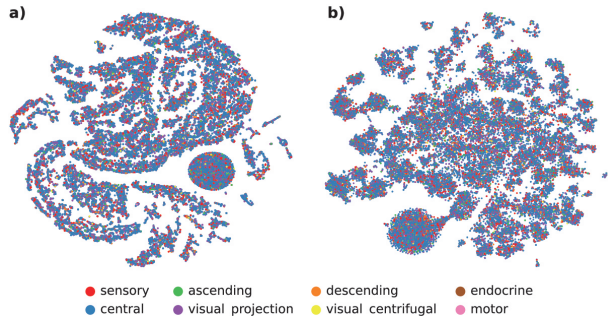

Figure 5. t-SNE clustering of neurons using **a)** spectral and **b)** Node2Vec features. Neurons are colored by their superclasses.

III. Exploitation

A. Methods

In the exploitation stage of the project, we aim to predict (i) the neuron superclass, (ii) the neuron class, and (iii) the hemilineage. We further subsume all superclasses and classes with fewer than 100 instances into a single "Other" category, along with neurons without superclass/class labels. These neurons are not considered in the classification task. We perform the same operation to hemilineage labels but with a threshold of 200. This leaves us with 7 superclasses, 15 classes, and 30 hemilineages. We separately evaluate the classification task performance using (i) Node2Vec features, (ii) spectral features, and (iii) both. The morphological features of the neuron (skeleton length, surface area, volume) are appended to the feature vector in all configurations.

We evaluated two non-GNN baseline models and two GNN models for the node classification tasks. The architecture of these models are shown in Fig. 6:

- A Logistic Regression model (**LR**).
- A Multi-layer Perceptron (**MLP**) with two hidden layers of size 16 and ReLU activation.
- A Graph Convolutional Network (**GCN**) [8] with three convolution layers with 16 hidden channels and ReLU activation.
- A model based on the Fast-Parapred (**FP**) model described in [9]. Our FP model uses a graph attention layer [10] on the output of the convolution layer before adding the outputs of both layers and passing the sum to a dense layer. This model is described in more details in the next subsection.

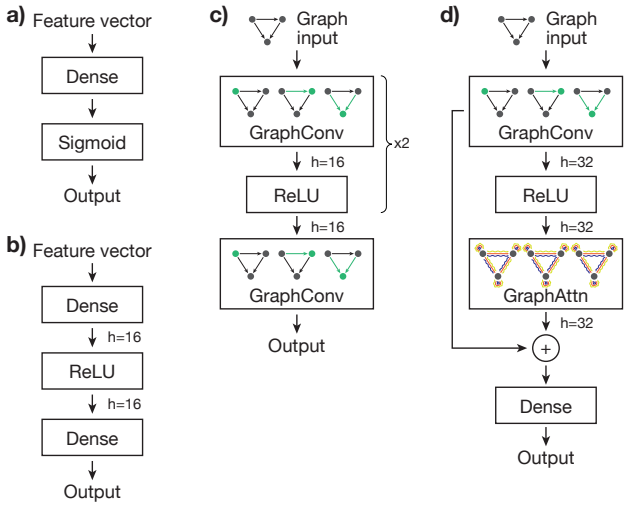

Figure 6. Schematic overview of the model architectures: **a)** Logistic Regression (LR), **b)** Multi-layer Perceptron (MLP), **c)** Graph Convolutional Network (GCN), **d)** Fast-Parapred (FP).

We split the dataset into training (60%), validating (20%), and testing (20%) sets. The GNN models are trained for 10,000 epochs, and the model parameters at the epoch with the highest validation F1 score are selected. We use the Adam optimizer [11] on the cross-entropy loss at a learning rate of 0.01.

B. Fast-Parapred Model

We build up on the developments of [9], in which their classification process involves a stack of three a trous (dilated) convolutional layers to extract features, followed by a self-attention mechanism to capture feature relationships. Finally, a pointwise fully-connected layer is

utilized to classify each antibody amino acid as binding or non-binding. They used the ELU (Exponential Linear Unit) activation function for the intermediate layers and the logistic Sigmoid function for binary classification.

Since our task requires less complex models (which will be clear in the following results section), we decided to keep only one convolutional layer with 32 features and no dilation. We also adopted the more classical ReLU activation function. To put this in more formal terms, the convolutional layer we are applying has the following math formulation:

$$h^{l+1} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{W}\tilde{D}^{-\frac{1}{2}}h^{l}\theta^{l+1}) \tag{2}$$

where W is the adjacency (weight) matrix, D is the degree matrix and θ are the learned parameters.
The convolved embeddings passed to the attention layer, which uses the *GATv2* layer from [12] paper. GATv2 further improves upon the GAT architecture by introducing several enhancements and modifications. These improvements aim to enhance the model's ability to capture and process information from graph-structured data. Very briefly, in this model, every node can attend to any other node, and the updated embedding h_i^{l+1} is given by:

$$h_i^{l+1} = \alpha_{ii}^{l}\theta h_i + \sum_{j \in N(i)} \alpha_{ij}\theta h_j \tag{3}$$

where $N(i)$ is the neighborhood of node i, θ are again the learnable parameters, and α_{ij} are the attention coefficients computed as:

$$\alpha_{ij} = \frac{\exp(a^T \text{LeakyReLU}(\theta[h_i \parallel h_j]))}{\sum_{k \in N(i)} \exp(a^T \text{LeakyReLU}(\theta[h_i \parallel h_k]))} \tag{4}$$

where a represents a separate learnable weight vector used in the computation of attention coefficients, alongside the parameter vector θ, and $\parallel$ is the concatenation operator.

In Fig.7 we show how promising this model is by looking at the first 1,000 epochs. In order to reach the results shown in the next section, 10,000 epochs are calculated and tested on validation, then we detached the estimated parameters corresponding to the epoch with the highest F1 score (which is safer compared to accuracy in terms of overfitting to validation data). Then the classificator is tested on unseen test data.

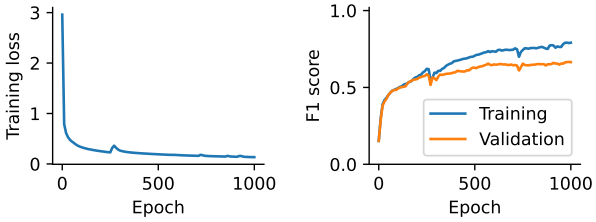

Figure 7. Training history for the FP model on the super_class classification task. First 1000 epochs

C. Results

The accuracies and the F1 scores of the models in all classification tasks are shown in Table I (the accuracies are shown as percentages; the F1 scores are shown in parentheses). The best performing models are highlighted in bold on each row. In particular, the accuracies of the models using both Node2Vec and spectral features are shown in Fig. 8.

Table I
MODEL PERFORMANCE SUMMARY

Task	Features	LR	MLP	GCN	FP
Superclass prediction	Node2Vec	82% (0.51)	89% (0.60)	89% (0.55)	**90% (0.66)**
	Spectral	76% (0.34)	86% (0.49)	**89%** (0.58)	89% **(0.62)**
	Both	84% (0.56)	90% (0.59)	89% (0.62)	**90% (0.65)**
Class prediction	Node2Vec	89% (0.81)	91% (0.82)	90% (0.83)	**92% (0.86)**
	Spectral	73% (0.53)	89% (0.80)	**92%** **(0.87)**	91% (0.84)
	Both	90% (0.84)	91% (0.84)	90% (0.84)	**92% (0.87)**
Hemilineage prediction	Node2Vec	47% (0.53)	48% (0.55)	53% (0.57)	**57% (0.63)**
	Spectral	30% (0.27)	41% (0.41)	52% **(0.57)**	**53%** (0.56)
	Both	49% (0.56)	49% (0.57)	54% (0.58)	**60% (0.66)**

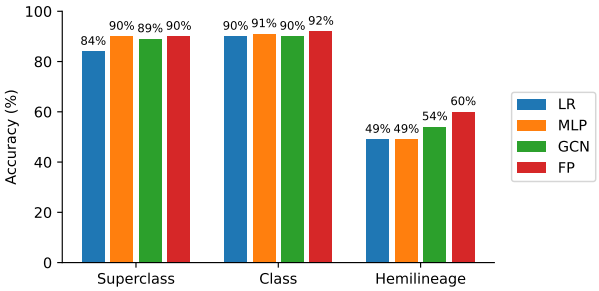

Figure 8. Performance of different models using both Node2Vec and spectral features on all three classification tasks. LR: Linear Regressor; MLP: Multi-layer Perceptron; GCN: Graph Convolutional Network; FP: Fast-Parapred.

The first notable finding was that all models showcased superior performance when utilizing node2vec features compared to spectral features. Node2vec, a widely used algorithm for learning node embeddings in graph-based data, appeared to capture crucial information from the graph structure that contributed to improved classification accuracy.

To explore the potential benefits of combining different feature sets, the study investigated the integration of node2vec and spectral features. Surprisingly, the improvement achieved by combining the two feature types was relatively small compared to using node2vec features alone. This suggests that the node2vec features already encompassed the most important information required for accurate classification.

Among the models evaluated, the FP (Feature Propagation) model emerged as the best performer. It consistently achieved the highest accuracy and F1 score across the tested tasks when using both embedding types. However, even the baseline models demonstrated promising results when utilizing node2vec features. The success of the baseline models showcases the importance of carefully selecting and engineering relevant features in machine learning tasks. It's true however that the difference in performance with more complex structures becomes more relevant when dealing with the hardest task, hemilineage classification, proving that the increased complexity allowed FP and GCN models to capture intricate patterns and dependencies within the data, leading to enhanced classification accuracy.

To gain further insights into the performance of the FP-Model, we can analyze the Confusion Matrices depicted in Figure 9. Notably, the counts within the matrices are presented in a log-scale format, and as usual diagonal elements represent the correctly predicted samples. As indicated by the lighter colors along the diagonal, the FP-Model demonstrates favorable performance across all three classification tasks.

However, upon closer examination, lighter colors are also visible outside the main diagonal, particularly in the superclass and hemilineage prediction tasks. These off-diagonal light colors indicate serious misclassifications made by the model. In particular, the hemilineage prediction task appears to exhibit more prominent misclassifications, with the top left corner of matrix **c** (the green square) showing noticeably lighter shades. This specific region corresponds to classes MBp1, MBp2, MBp3, and MBp4, suggesting that the model struggles to distinguish these classes accurately.

To enhance the model's performance in the hemilineage prediction task, it is crucial to address the challenges associated with differentiating classes MBp1, MBp2, MBp3, and MBp4. Focusing on refining the model's predictive results for these specific classes can potentially lead to improved accuracy and more precise classification of the task.

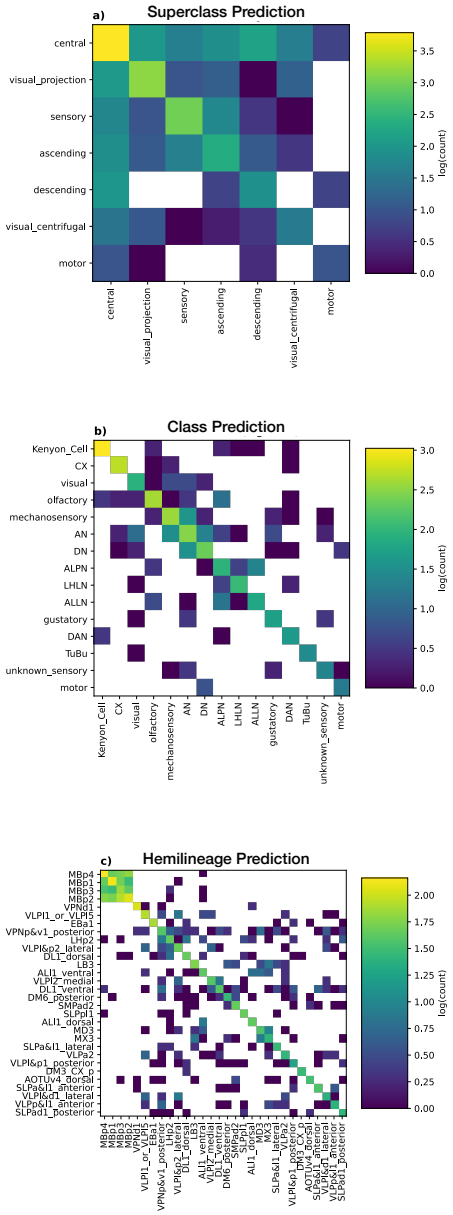

Figure 9. Confusion matrices on test data using the FP model: **a)** superclass prediction, **b)** class prediction, **c)** hemilineage prediction.

D. Discussion

One limitation of the study is the incomplete labeling of neuron classifications and hemilineages in the dataset. The prediction models are trained on the available labeled data, which may introduce biases and limit the generalizability of the results. Obtaining more complete and accurate labels would enhance the reliability of the predictions. On top of this, including optic neurons and improving their proofreading could provide a more comprehensive understanding of the brain's wiring diagram.

The findings of this study provide insights into the predictability of neuron types from their connectivity patterns in the *Drosophila* brain. While the specific results are applicable to the fruit fly connectome, the methodology and approach can be extended to other organisms and brain regions. By adapting the graph embedding and prediction models to different datasets, it is possible to uncover common principles underlying brain organization and function.

IV. Conclusion

In this project, we showed that information about the neuron (in particular the neuron type and the hemilineage) can be reliably predicted from the structural wiring diagram of the brain. This is made especially effective with novel techniques such as Node2Vec embedding, graph convolution, and graph attention layers. We hope that this work will serve as a proof-of-concept for simplifying the task of assigning neuron properties in future connectomics datasets.

ACKNOWLEDGEMENTS

We thank the FlyWire community (https://flywire.ai/) for granting us early access to its unpublished dataset.

REFERENCES

[1] S. Dorkenwald, C. E. McKellar, T. Macrina *et al.*, "FlyWire: online community for whole-brain connectomics," *Nature Methods*, vol. 19, no. 1, pp. 119–128, Dec. 2021. [Online]. Available: https://doi.org/10.1038/s41592-021-01330-0

[2] L. K. Scheffer, C. S. Xu, M. Januszewski *et al.*, "A connectome and analysis of the adult drosophila central brain," *eLife*, vol. 9, Sep. 2020. [Online]. Available: https://doi.org/10.7554/elife.57443

[3] Z. Zheng, J. S. Lauritzen, E. Perlman *et al.*, "A complete electron microscopy volume of the brain of adult drosophila melanogaster," *Cell*, vol. 174, no. 3, pp. 730–743.e22, Jul. 2018. [Online]. Available: https://doi.org/10.1016/j.cell.2018.06.019

[4] J. C. Eccles, P. Fatt, and K. Koketsu, "Cholinergic and inhibitory synapses in a pathway from motor-axon collaterals to motoneurones," *The Journal of Physiology*, vol. 126, no. 3, pp. 524–562, Dec. 1954. [Online]. Available: https://doi.org/10.1113/jphysiol.1954.sp005226

[5] F. Chung, "Laplacians and the cheeger inequality for directed graphs," *Annals of Combinatorics*, vol. 9, no. 1, pp. 1–19, Apr. 2005. [Online]. Available: https://doi.org/10.1007/s00026-005-0237-z

[6] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, Aug. 2016. [Online]. Available: https://doi.org/10.1145/2939672.2939754

[7] L. van der Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. 86, pp. 2579–2605, 2008. [Online]. Available: http://jmlr.org/papers/v9/vandermaaten08a.html

[8] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016.

[9] A. Deac, P. VeliČković, and P. Sormanni, "Attentive cross-modal paratope prediction," *Journal of Computational Biology*, vol. 26, no. 6, pp. 536–545, Jun. 2019. [Online]. Available: https://doi.org/10.1089/cmb.2018.0175

[10] P. Veličković, G. Cucurull, A. Casanova *et al.*, "Graph attention networks," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=rJXMpikCZ

[11] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014.

[12] S. Brody, U. Alon, and E. Yahav, "How attentive are graph attention networks?" *arXiv preprint arXiv:2105.14491*, 2021.